A METHOD FOR SETTING UP A SECURITY ASSOCIATION

The field of the invention

[0001] The present invention relates to a method for setting up a security association.

BACKGROUND TO THE INVENTION

[0002] A communication system can be seen as a facility that enables communication between two or more entities such as user equipment and/or other nodes associated with the system. The communication may comprise, for example, communication of voice, data, multimedia and so on.

[0003] A communication system typically operates in accordance with a given standard or specification which sets out what the various elements of the system are permitted to do and how that should be achieved. For example, the standard or specification may define if the user, or more precisely, user equipment or terminal is provided with a circuit switched service and/or a packet switched service. Communication protocols and/or parameters which shall be used for the connection may also be defined. In other words, a specific set of "rules" on which the communication can be based on needs to be defined to enable communication by means of the system.

[0004] Communication systems proving wireless communication for user terminals or other nodes are known. An example of the wireless systems is a cellular network. In cellular systems, a base transceiver station (BTS) or similar access entity serves mobile stations (MS) or similar user equipment (UE) via a wireless interface between these entities. The operation of the apparatus required for the communication can be controlled by one or several control entities. The various control entities may be interconnected. One or more

gateway nodes may also be provided for connecting the cellular network to other networks, such as to another cellular system or to a public switched telephone network (PSTN) and/or other communication networks such as an IP (Internet Protocol) and/or other packet switched networks. The communication between the user equipment and the elements of the communication network can be based on an appropriate communication protocol such as the session initiation protocol (SIP).

[0005] For example, in the current third generation (3G) multimedia network architectures it is assumed that various servers are used for handling different functions. These include functions such as call state control functions (CSCFs). A call state control function entity may provide functions such as proxy call state control (P-CSCF), interrogating call state control (I-CSCF), and serving call state control (S-CSCF). The serving call state control function can be divided further between originating call state control function (O-CSCF) and terminating call state control function (T-CSCF) at the originating and terminating ends of a session, respectively. Control functions may also be provided by entities such as a home subscriber server (HSS) and various application servers.

[0006] It shall be appreciated that the term "session" refers to any communication a user may have such as to a call, data (e.g. web browsing) or multimedia communication and so on.

[0007] The wireless communication network and internet network technology are gradually converging to make the packet switched data services used in the internet available to mobile users. Initially, the technology developed for the internet has primarily been designed for desk top computers and medium to high band width data connections. In contrast, the mobile terminal environments are generally characterised by less band width and smaller connection stability in comparison to the fixed networks. Additionally, terminals have tended to have smaller displays, less memory and less powerful processors as compared to desk top computers or the like.

[0008] However, IP (internet protocol) base packet services which are usable in a wireless mobile environment are being developed at an increasing rate. This is partly due to demand by users of mobile terminals and partly due to the development of new technologies which are attempting to increase the available band width, improve quality of service and data security. The new standards which are being developed include GPRS (general packet radio service), UMTS (universal mobile telecommunications system) and WLAN (Wireless Local Area Network). These are by way of example only. The GPRS standard aims to provide high quality services for GSM subscribers by efficiently using the GSM infrastructure and protocols. In addition, the GPRS radio services is designed to provide packet based services. WLAN standards (e.g. IEEE 802.11 and HiperLAN) define a Ethernet-like network that uses radio waves instead of cables.

[0009] One issue which needs to be addressed in communication networks relates to security and mobility. Mobility, for example when a mobile station moves from one cell to the next or from one network to another, or even from one service provider to another, requires the address of the mobile station to change. On the other hand, security requires the address of the mobile station to be identified reliably.

[0010] Reference is made to the IETF (internet engineering task force) standard RFC2401 which sets out the security architecture for the internet protocol. This standard is hereby incorporated by reference. In this standard, various security services for traffic at the IP (internet protocol) layer both in IPv4 and IPv6 environments is set out. IPSec or IP security defined in this IETF standard is designed to provide interoperable, high quality, cryptographically based security for IPv4 and IPv6. The set of security services offered include access control, connectionless integrity, data origin authentication, protection against replays, confidentiality (ie encryption) and limited traffic flow confidentiality. These services are provided at the IP layer.

[0011] Reference is made to a third generation IMS domain. During the registration procedure the security association SA parameters are exchanged in order to establish an IPSec SA (internet protocol security association). If the SA selectors are set to the actual IP address of the UE user equipment, when the UE generates a new IP address it will need to re-register and re-establish the SA. This is far too complicated and need to be simplified. Reference is made to Figure 2 which illustrates this. The user equipment 100 has an IP address and associated with that one IP address a security association 102. The security association 102 is between the user equipment 100 and proxy CSCF 104. When the user equipment changes its IP address as will happen from time to time, a new security association needs to be established.

[0012] It has been proposed that when a UE changes its IP address, e.g. by using the method described in RFC 3041 which is attached as annexe A, then the UE shall delete the existing SA and initiate an unprotected registration procedure using the new IP address as the source IP address in the packets carrying the REGISTER messages. This however has the disadvantage that this complicates the SA management in the P-CSCF, as there might be ongoing sessions etc. when the need may arise at the UE to delete the current SA and set up a new one.

SUMMARY OF THE INVENTION

[0013] It is an aim of embodiments of the present invention to address the problem discussed above.

[0014] Aspects of the invention can be seen from the attached claims.

[0015] Embodiments of the invention may use an additional parameter to be exchanged during the registration procedure. The parameter may be part of the Security-Client header field defined in draft-ietf-sip-sec-agree (which is incorporated as Annexe B)

BRIEF DESCRIPTION OF DRAWINGS

[0016] For a better understanding of the present invention and as to how the same may be carried into effect, reference will now be made by way of example only to the accompanying drawings in which:

[0017] Figure 1 shows schematically a system with which embodiments of the present invention may be used;

[0018] Figure 2 illustrates schematically the prior art;

[0019] Figure 3 illustrates schematically an embodiment of the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0020] Reference is first made to Figure 1 which shows a possible network architecture wherein the present invention may be embodied. The exemplifying network system 10 is arranged in accordance with UMTS 3G specifications. The cellular system 10 is divided between a radio access network (RAN) 2 and a core network (CN).

[0021] In general terms, it is possible to describe a communication system as a model in which the functions of the system are divided in several hierarchically arranged function layers. Figure 1 shows three different function layers, i.e. a service layer, an application layer and a transport layer and the positioning of various network elements relative to these layers. It shall be appreciated that the layered model is shown only in order to illustrate the relationships between the various functions of a data communication system. In a physical i.e. real implementation the entities (e.g. servers or other nodes) are typically not arranged in a layered manner.

[0022] A plurality of user equipment 1 is served by a 3G radio access network (RAN) 2 over a wireless interface. The user equipment is enabled to move relative to the access entity, and may thus be referred to by the term mobile station. The radio access network function is hierarchically located on the

transport layer. It shall be appreciated that although Figure 1 shows only one radio access network for clarity reasons, a typical communication network system comprises a number of radio access networks.

[0023] The 3G radio access network (RAN) 2 is shown to be physically connected to a serving general packet radio service support node (SGSN) entity 3. The SGSN 3 is a part of the core network. In the functional model the entity 3 belongs to the transport layer. The operation of a typical cellular network and the various transport level entities thereof is known by the skilled person and will thus not be explained in more detail herein.

[0024] An application layer 20 is shown to be located on top of the transport layer. The application layer 20 may include several application level functions. Figure 1 shows two call state control entities (CSCFs) 22 and 23. From these the call state server 22 is the so called serving call state control function (S-CSCF) wherein the user equipment 1 is registered to. That is, the server 22 is currently serving said user equipment 1 and is in control of the status of said user equipment.

[0025] The application layer is also shown to comprise a home subscriber server (HSS) entity 24. The home subscriber server (HSS) 24 is for storing data such as the registration identifiers (ID), their status (currently-registered-with-S-CSCF1 or currently-not-registered) and similar user related information.

[0026] For the sake of completeness some other elements such as various gateway entities (e.g. the Media Gateway Control Function MGCF, Media Gateway MGW and the Signalling Gateway SGW) are also shown. However, these do not form an essential part of the invention and will thus not be described in more detail.

[0027] The solid lines indicate actual data communication between various entities. The dashed lines indicate signalling traffic between various entities. The signalling is typically required for management and/or control functions, such as

for registration, session set-up, charging and so on. As can be seen, user equipment 1 may have communication via the access network 2 and appropriate gateways with various other networks such as networks 4, 5 and 6. The other networks may be adapted to operate in accordance with the same standard as the network 10 or any other appropriate standard.

[0028] Reference is made to Figure 3 which shows the embodiment of the invention, In this arrangement the user equipment 100 has a security association with the security association 106. The security association 106 is between the user equipment and the proxy CSCF 104. In this embodiment of the invention, a range of IP addresses are associated with the security association.

[0029] The UE sends a REGISTER request towards the IMS network.

REGISTER sip:registrar.home1.net SIP/2.0
Via: SIP/2.0/UDP [5555::aaa:bbb:ccc:ddd];branch=z9hG4bKnashds7
Max-Forwards: 70
From: <sip:user1_public1@home1.net>;tag=4fa3
To: <sip:user1_public1@home1.net>
Contact: sip:[5555::aaa:bbb:ccc:ddd]
Call-ID: apb03a0s09dkjdfglkj49111
Authorization: Digest username="user1_private@home1.net", realm="registrar.home1.net", nonce="", uri="sip:registrar.home1.net", response=""
Security-Client: ipsec-man; alg=HMAC-SHA1; SPI_U_UDP=12345678; SPI_U_TCP=23456789; Port_U_UDP=1357; Port_U_TCP=1358; **addr-pref-length=64**
Require: sec-agree
CSeq: 1 REGISTER
Expires: 7200
Content-Length: 0

[0030] The prefix parameter is added into the Security-Client header and specifies the UE's wish to use a 64 bit prefix for the outgoing SA. Once the P-CSCF reads the header it will instruct the IPSec engine in the P-CSCF to set up the SAs towards the IPv6 prefix of the UE (e.g. 1234:abcd:5678::). In embodiments of the present the invention, the prefix length (also known as Subnet Mask in IPv4) the UE has got allocated is communicated, with the P-CSCF. The P-CSCF will then set up the SA towards the prefix of the UE's address, which will enable the UE to generate new IP addresses (within the prefix) for itself and send packets with that address inside the SA.

[0031] Embodiments of the present invention propose to use a new parameter in the Security-Client header. The name of the prefix may be 'addr-pref-length', and is inserted into the above-mentioned SIP header together with the other parameters needed for security association setup.

[0032] For a 3GPP IMS terminal the value of that parameter could either take the value 128 (meaning that the UE would like to have the outgoing SA towards towards the IMS network from its fixed IP address – i.e. the UE would not implement RFC3041 and would not make use of its benefits) or 64 (meaning that the UE would like to have the outgoing SA towards the IMS network from its prefix which was allocated to it by the GGSN). Once the UE specifies a prefix of 64 bits towards the P-CSCF, it may generate new IP addresses and send packets towards the IMS network with the newly generated IP addresses protected in the existing SA.

[0033] The addresses of the IP addresses associated with a security association can be of any suitable form. For example a range of addresses can be defined or information as to which addresses are usable or the like.

[0034] Thus the prefix value is either 128 (RFC3041 not used, SA for one IP address only) or a smaller value, preferably between 4 and 64 (RFC3041 is used, SA for a range of IP addresses) referring to a portion of the IP address that

does not change. The prefix may form part of the address and an address containing that prefix may be used with the defined security association. The information provided by the prefix may be used to provide information to define the range of addresses. Other methods of defining the usable addresses may be used in embodiments of the invention.

[0035] In further development, the value of this parameter could take an arbitrary value, but for a 3GPP Rel5 compliant terminal only these two values are valid.

[0036] It should be appreciated that the values are by way of example and can take any other suitable form.

[0037] The solution described above would not require the UE to re-register each time it generates a new IP address. As a consequence, the UE would not be required to delete the existing SA and set up a new one towards its newly generated IP address.

[0038] In embodiments of the invention, the IPv6 prefix of the UE in the SA selectors is used instead of the fixed IP address. It is preferred but not essential that the choice of the selector be left to the UE, i.e. the UE will communicate its wish to the P-CSCF (IMS network).

[0039] The parameter embodying the invention is preferably part of the Security-Client header, those syntax allows for new parameters to be added there.

[0040] The UE is arranged to insert this parameter into the Security-Client header. If the parameter is present, the P-CSCF would instruct the IPSec engine to set up the SA either for the fixed IP address of the mobile or the prefix calculated from the fixed IP address (based on the value of the parameter).

[0041] Embodiments of the invention are such that the UE is able to send packets with different source IP addresses protected in the same SA.

[0042] One advantage of embodiments of the present invention is that when the UE changes its IP address using RFC3041 auto-configuration, there is no need to re-register (in order to create a new security association) with the proxy as the security association is valid for a range of IP addresses instead of one IP address only.

[0043] Without embodiments of the invention, any ongoing sessions should be dropped, and besides re-registration (which is not the only problem here), a new security association needs to be negotiated every time the UE performs auto-configuration.

[0044] If the proxy is informed about the prefix length that the UE uses, the SA can be agreed accordingly. The prefix itself is allocated by the GGSN and it is unique for the UE. Note, when prefix length equals 128 is indicated to the proxy, then only one IP address (the full current IP address) is used for the security association (i.e. no RFC3041 auto-configuration is implemented in UE). In this latter case, re-registration and new SA is needed when IP address changes.

The IETF (internet engineering task force) draft standard RFC 3041 incorporated as Annexe A below. This draft describes how to create an IP address consisting of a prefix and an interface identifier (IPv6 address is 128 bits total, prefix is usually 64, the rest is interface ID freely generated by the UE). In the IP Multimedia Subsystem (IMS), where embodiments of the invention may be implemented, the prefix is allocated by the GGSN (an access router in general), while the interface ID is generated by the UE. The UE is allowed to create new IP addresses by generating and adding new interface IDs to the prefix. The problem arises when there is no valid security association between the UE and the P-CSCF for the new IP addresses. The UE can use any address for communication, but only the address(es) with a valid security association will reach the P-CSCF, which is the first contact point towards the IMS. When the communication fails at the first contact point, a new registration is required using

the new IP address. Any ongoing communication using the previous IP address will be released

Annex A

[0045] Privacy Extensions for Stateless Address Autoconfiguration in IPv6

Status of this Memo

[0046] This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

[0047] Nodes use IPv6 stateless address autoconfiguration to generate addresses without the necessity of a Dynamic Host Configuration Protocol (DHCP) server. Addresses are formed by combining network prefixes with an interface identifier. On interfaces that contain embedded IEEE Identifiers, the interface identifier is typically derived from it. On other interface types, the interface identifier is generated through other means, for example, via random number generation. This document describes an extension to IPv6 stateless address autoconfiguration for interfaces whose interface identifier is derived from an IEEE identifier. Use of the extension causes nodes to generate global-scope addresses from interface identifiers that change over time, even in cases where the interface contains an embedded IEEE identifier. Changing the interface identifier (and the global-scope addresses generated from it) over time makes it more difficult for eavesdroppers and other information collectors to identify when different addresses used in different transactions actually correspond to the same node.

1. Introduction

[0048] Stateless address autoconfiguration [ADDRCONF] defines how an IPv6 node generates addresses without the need for a DHCP server. Some types of network interfaces come with an embedded IEEE Identifier (i.e., a link-layer MAC address), and in those cases stateless address autoconfiguration uses the IEEE identifier to generate a 64- bit interface identifier [ADDRARCH]. By design, the interface identifier is likely to be globally unique when generated in this fashion. The interface identifier is in turn appended to a prefix to form a 128-bit IPv6 address.

[0049] All nodes combine interface identifiers (whether derived from an IEEE identifier or generated through some other technique) with the reserved link-local prefix to generate link-local addresses for their attached interfaces. Additional addresses, including site-local and global-scope addresses, are then created by combining prefixes advertised in Router Advertisements via Neighbor Discovery [DISCOVERY] with the interface identifier.

[0050] Not all nodes and interfaces contain IEEE identifiers. In such cases, an interface identifier is generated through some other means (e.g., at random), and the resultant interface identifier is not globally unique and may also change over time. The focus of this document is on addresses derived from IEEE identifiers, as the concern being addressed exists only in those cases where the interface identifier is globally unique and non-changing. The rest of this document assumes that IEEE identifiers are being used, but the techniques described may also apply to interfaces with other types of globally unique and/or persistent identifiers. This document discusses concerns associated with the embedding of non-changing interface identifiers within IPv6 addresses and describes extensions to stateless address autoconfiguration that can help mitigate those concerns for individual users and in environments where such concerns are significant. Section 2 provides background information on the issue. Section 3 describes a procedure for generating alternate interface identifiers and global-scope addresses. Section 4 discusses implications of changing interface identifiers.

2. Background

[0051] This section discusses the problem in more detail, provides context for evaluating the significance of the concerns in specific environments and makes comparisons with existing practices.

2.1. Extended Use of the Same Identifier

[0052] The use of a non-changing interface identifier to form addresses is a specific instance of the more general case where a constant identifier is reused over an extended period of time and in multiple independent activities. Anytime the same identifier is used in multiple contexts, it becomes possible for that identifier to be used to correlate seemingly unrelated activity. For example, a network sniffer placed strategically on a link across which all traffic to/from a particular host crosses could keep track of which destinations a node communicated with and at what times. Such information can in some cases be used to infer things, such as what hours an employee was active, when someone is at home, etc.

[0053] One of the requirements for correlating seemingly unrelated activities is the use (and reuse) of an identifier that is recognizable over time within different contexts. IP addresses provide one obvious example, but there are more. Many nodes also have DNS names associated with their addresses, in which case the DNS name serves as a similar identifier. Although the DNS name associated with an address is more work to obtain (it may require a DNS query) the information is often readily available. In such cases, changing the address on a machine over time would do little to address the concerns raised in this document, unless the DNS name is changed as well (see Section 4).

[0054] Web browsers and servers typically exchange "cookies" with each other [COOKIES]. Cookies allow web servers to correlate a current activity with a previous activity. One common usage is to send back targeted advertising to a user by using the cookie supplied by the browser to identify what earlier queries had been made (e.g., for what type of information). Based on the earlier

queries, advertisements can be targeted to match the (assumed) interests of the end-user.

[0055] The use of a constant identifier within an address is of special concern because addresses are a fundamental requirement of communication and cannot easily be hidden from eavesdroppers and other parties. Even when higher layers encrypt their payloads, addresses in packet headers appear in the clear. Consequently, if a mobile host (e.g., laptop) accessed the network from several different locations, an eavesdropper might be able to track the movement of that mobile host from place to place, even if the upper layer payloads were encrypted [SERIALNUM].

2.2. Address Usage in IPv4 Today

[0056] Addresses used in today's Internet are often non-changing in practice for extended periods of time, especially in non-home environments (e.g., corporations, campuses, etc.). In such sites, addresses are assigned statically and typically change infrequently. Over the last few years, sites have begun moving away from static allocation to dynamic allocation via DHCP [DHCP]. In theory, the address a client gets via DHCP can change over time, but in practice servers often return the same address to the same client (unless addresses are in such short supply that they are reused immediately by a different node when they become free). Thus, even within sites using DHCP, clients frequently end up using the same address for weeks to months at a time.

[0057] For home users accessing the Internet over dialup lines, the situation is generally different. Such users do not have permanent connections and are often assigned temporary addresses each time they connect to their ISP (e.g., AOL). Consequently, the addresses they use change frequently over time and are shared among a number of different users. Thus, an address does not reliably identify a particular device over time spans of more than a few minutes.

[0058] A more interesting case concerns always-on connections (e.g., cable modems, ISDN, DSL, etc.) that result in a home site using the same address for

extended periods of time. This is a scenario that is just starting to become common in IPv4 and promises to become more of a concern as always-on internet connectivity becomes widely available. Although it might appear that changing an address regularly in such environments would be desirable to lessen privacy concerns, it should be noted that the network prefix portion of an address also serves as a constant identifier. All nodes at (say) a home, would have the same network prefix, which identifies the topological location of those nodes. This has implications for privacy, though not at the same granularity as the concern that this document addresses. Specifically, all nodes within a home would be grouped together for the purposes of collecting information. This issue is difficult to address, because the routing prefix part of an address contains topology information and cannot contain arbitrary values.

[0059] Finally, it should be noted that nodes that need a (non-changing) DNS name generally have static addresses assigned to them to simplify the configuration of DNS servers. Although Dynamic DNS [DDNS] can be used to update the DNS dynamically, it is not yet widely deployed. In addition, changing an address but keeping the same DNS name does not really address the underlying concern, since the DNS name becomes a non-changing identifier. Servers generally require a DNS name (so clients can connect to them), and clients often do as well (e.g., some servers refuse to speak to a client whose address cannot be mapped into a DNS name that also maps back into the same address). Section 4 describes one approach to this issue.

2.3. The Concern With IPv6 Addresses

[0060] The division of IPv6 addresses into distinct topology and interface identifier portions raises an issue new to IPv6 in that a fixed portion of an IPv6 address (i.e., the interface identifier) can contain an identifier that remains constant even when the topology portion of an address changes (e.g., as the result of connecting to a different part of the Internet). In IPv4, when an address changes, the entire address (including the local part of the address) usually changes. It is this new issue that this document addresses.

[0061] If addresses are generated from an interface identifier, a home user's address could contain an interface identifier that remains the same from one dialup session to the next, even if the rest of the address changes. The way PPP is used today, however, PPP servers typically unilaterally inform the client what address they are to use (i.e., the client doesn't generate one on its own). This practice, if continued in IPv6, would avoid the concerns that are the focus of this document.

[0062] A more troubling case concerns mobile devices (e.g., laptops, PDAs, etc.) that move topologically within the Internet. Whenever they move (in the absence of technology such as mobile IP [MOBILEIP]), they form new addresses for their current topological point of attachment. This is typified today by the "road warrior" who has Internet connectivity both at home and at the office. While the node's address changes as it moves, however, the interface identifier contained within the address remains the same (when derived from an IEEE Identifier). In such cases, the interface identifier can be used to track the movement and usage of a particular machine [SERIALNUM]. For example, a server that logs usage information together with a source addresses, is also recording the interface identifier since it is embedded within an address. Consequently, any data-mining technique that correlates activity based on addresses could easily be extended to do the same using the interface identifier. This is of particular concern with the expected proliferation of next-generation network-connected devices (e.g., PDAs, cell phones, etc.) in which large numbers of devices are in practice associated with individual users (i.e., not shared). Thus, the interface identifier embedded within an address could be used to track activities of an individual, even as they move topologically within the internet.

[0063] In summary, IPv6 addresses on a given interface generated via Stateless Autoconfiguration contain the same interface identifier, regardless of where within the Internet the device connects. This facilitates the tracking of individual devices (and thus potentially users). The purpose of this document is

to define mechanisms that eliminate this issue, in those situations where it is a concern.

2.4. Possible Approaches

[0064] One way to avoid some of the problems discussed above is to use DHCP for obtaining addresses. With DHCP, the DHCP server could arrange to hand out addresses that change over time.

[0065] Another approach, compatible with the stateless address autoconfiguration architecture, would be to change the interface id portion of an address over time and generate new addresses from the interface identifier for some address scopes. Changing the interface identifier can make it more difficult to look at the IP addresses in independent transactions and identify which ones actually correspond to the same node, both in the case where the routing prefix portion of an address changes and when it does not.

[0066] Many machines function as both clients and servers. In such cases, the machine would need a DNS name for its use as a server. Whether the address stays fixed or changes has little privacy implication since the DNS name remains constant and serves as a constant identifier. When acting as a client (e.g., initiating communication), however, such a machine may want to vary the addresses it uses. In such environments, one may need multiple addresses: a "public" (i.e., non-secret) server address, registered in the DNS, that is used to accept incoming connection requests from other machines, and a "temporary" address used to shield the identity of the client when it initiates communication. These two cases are roughly analogous to telephone numbers and caller ID, where a user may list their telephone number in the public phone book, but disable the display of its number via caller ID when initiating calls.

[0067] To make it difficult to make educated guesses as to whether two different interface identifiers belong to the same node, the algorithm for generating alternate identifiers must include input that has an unpredictable component from the perspective of the outside entities that are collecting

information. Picking identifiers from a pseudo-random sequence suffices, so long as the specific sequence cannot be determined by an outsider examining information that is readily available or easily determinable (e.g., by examining packet contents). This document proposes the generation of a pseudo-random sequence of interface identifiers via an MD5 hash.

[0068] Periodically, the next interface identifier in the sequence is generated, a new set of temporary addresses is created, and the previous temporary addresses are deprecated to discourage their further use. The precise pseudo-random sequence depends on both a random component and the globally unique interface identifier (when available), to increase the likelihood that different nodes generate different sequences.

3. Protocol Description

[0069] The goal of this section is to define procedures that:

[0070] 1) Do not result in any changes to the basic behavior of addresses generated via stateless address autoconfiguration [ADDRCONF].

[0071] 2) Create additional global-scope addresses based on a random interface identifier for use with global scope addresses. Such addresses would be used to initiate outgoing sessions. These "random" or temporary addresses would be used for a short period of time (hours to days) and would then be deprecated. Deprecated address can continue to be used for already established connections, but are not used to initiate new connections. New temporary addresses are generated periodically to replace temporary addresses that expire, with the exact time between address generation a matter of local policy.

[0072] 3) Produce a sequence of temporary global-scope addresses from a sequence of interface identifiers that appear to be random in the sense that it is difficult for an outside observer to predict a future address (or identifier) based on

a current one and it is difficult to determine previous addresses (or identifiers) knowing only the present one.

[0073] 4) Generate a set of addresses from the same (randomized) interface identifier, one address for each prefix for which a global address has been generated via stateless address autoconfiguration. Using the same interface identifier to generate a set of temporary addresses reduces the number of IP multicast groups a host must join. Nodes join the solicited-node multicast address for each unicast address they support, and solicited-node addresses are dependent only on the low-order bits of the corresponding address.

[0074] This decision was made to address the concern that a node that joins a large number of multicast groups may be required to put its interface into promiscuous mode, resulting in possible reduced performance.

3.1. Assumptions

[0075] The following algorithm assumes that each interface maintains an associated randomized interface identifier. When temporary addresses are generated, the current value of the associated randomised interface identifier is used. The actual value of the identifier changes over time as described below, but the same identifier can be used to generate more than one temporary address.

[0076] The algorithm also assumes that for a given temporary address, an implementation can determine the corresponding public address from which it was generated. When a temporary address is deprecated, a new temporary address is generated. The specific valid and preferred lifetimes for the new address are dependent on the corresponding lifetime values in the public address.

[0077] Finally, this document assumes that when a node initiates outgoing communication, temporary addresses can be given preference over public addresses. This can mean that all connections initiated by the node use

temporary addresses by default, or that applications individually indicate whether they prefer to use temporary or public addresses. Giving preference to temporary address is consistent with on-going work that addresses the topic of source-address selection in the more general case [ADDR_SELECT]. An implementation may make it a policy that it does not select a public address in the event that no temporary address is available (e.g., if generation of a useable temporary address fails).

3.2. Generation Of Randomized Interface Identifiers.

[0078] We describe two approaches for the maintenance of the randomised interface identifier. The first assumes the presence of stable storage that can be used to record state history for use as input into the next iteration of the algorithm across system restarts. A second approach addresses the case where stable storage is unavailable and there is a need to generate randomized interface identifiers without previous state.

3.2.1. When Stable Storage Is Present

[0079] The following algorithm assumes the presence of a 64-bit "history value" that is used as input in generating a randomized interface identifier. The very first time the system boots (i.e., out-of-the-box), a random value should be generated using techniques that help ensure the initial value is hard to guess [RANDOM]. Whenever a new interface identifier is generated, a value generated by the computation is saved in the history value for the next iteration of the algorithm.

[0080] A randomized interface identifier is created as follows:

[0081] 1) Take the history value from the previous iteration of this algorithm (or a random value if there is no previous value) and append to it the interface identifier generated as described in [ADDRARCH].

[0082] 2) Compute the MD5 message digest [MD5] over the quantity created in the previous step.

[0083] 3) Take the left-most 64-bits of the MD5 digest and set bit 6 (the left-most bit is numbered 0) to zero. This creates an interface identifier with the universal/local bit indicating local significance only. Save the generated identifier as the associated randomized interface identifier.

[0084] 4) Take the rightmost 64-bits of the MD5 digest computed in step 2) and save them in stable storage as the history value to be used in the next iteration of the algorithm.

[0085] MD5 was chosen for convenience, and because its particular properties were adequate to produce the desired level of randomization. IPv6 nodes are already required to implement MD5 as part of IPsec [IPSEC], thus the code will already be present on IPv6 machines.

[0086] In theory, generating successive randomized interface identifiers using a history scheme as above has no advantages over generating them at random. In practice, however, generating truly random numbers can be tricky. Use of a history value is intended to avoid the particular scenario where two nodes generate the same randomised interface identifier, both detect the situation via DAD, but then proceed to generate identical randomized interface identifiers via the same (flawed) random number generation algorithm. The above algorithm avoids this problem by having the interface identifier (which will often be globally unique) used in the calculation that generates subsequent randomized interface identifiers. Thus, if two nodes happen to generate the same randomized interface identifier, they should generate different ones on the followup attempt.

3.2.2. In The Absence of Stable Storage

[0087] In the absence of stable storage, no history value will be available across system restarts to generate a pseudo-random sequence of interface identifiers. Consequently, the initial history value used above will need to be generated at random. A number of techniques might be appropriate. Consult [RANDOM] for suggestions on good sources for obtaining random numbers. Note that even though machines may not have stable storage for storing a

history value, they will in many cases have configuration information that differs from one machine to another (e.g., user identity, security keys, serial numbers, etc.). One approach to generating a random initial history value in such cases is to use the configuration information to generate some data bits (which may remain constant for the life of the machine, but will vary from one machine to another), append some random data and compute the MD5 digest as before.

3.3. Generating Temporary Addresses

[0088] [ADDRCONF] describes the steps for generating a link-local address when an interface becomes enabled as well as the steps for generating addresses for other scopes. This document extends [ADDRCONF] as follows. When processing a Router Advertisement with a Prefix Information option carrying a global-scope prefix for the purposes of address autoconfiguration (i.e., the A bit is set), perform the following steps:

[0089] 1) Process the Prefix Information Option as defined in [ADDRCONF], either creating a public address or adjusting the lifetimes of existing addresses, both public and temporary. When adjusting the lifetimes of an existing temporary address, only lower the lifetimes. Implementations must not increase the lifetimes of an existing temporary address when processing a Prefix Information Option.

[0090] 2) When a new public address is created as described in [ADDRCONF] (because the prefix advertised does not match the prefix of any address already assigned to the interface, and the Valid Lifetime in the option is not zero), also create a new temporary address.

[0091] 3) When creating a temporary address, the lifetime values are derived from the corresponding public address as follows:

[0092] - Its Valid Lifetime is the lower of the Valid Lifetime of the public address or TEMP_VALID_LIFETIME.

[0093] - Its Preferred Lifetime is the lower of the Preferred Lifetime of the public address or TEMP_PREFERRED_LIFETIME -DESYNC_FACTOR.

[0094] A temporary address is created only if this calculated Preferred Lifetime is greater than REGEN_ADVANCE time units. In particular, an implementation must not create a temporary address with a zero Preferred Lifetime.

[0095] 4) New temporary addresses are created by appending the interface's current randomized interface identifier to the prefix that was used to generate the corresponding public address. If by chance the new temporary address is the same as an address already assigned to the interface, generate a new randomized interface identifier and repeat this step.

[0096] 5) Perform duplicate address detection (DAD) on the generated temporary address. If DAD indicates the address is already in use, generate a new randomized interface identifier as described in Section 3.2 above, and repeat the previous steps as appropriate up to 5 times. If after 5 consecutive attempts no non-unique address was generated, log a system error and give up attempting to generate temporary addresses for that interface.

[0097] Note: because multiple temporary addresses are generated from the same associated randomized interface identifier, there is little benefit in running DAD on every temporary address. This document recommends that DAD be run on the first address generated from a given randomized identifier, but that DAD be skipped on all subsequent addresses generated from the same randomized interface identifier.

3.4. Expiration of Temporary Addresses

[0098] When a temporary address becomes deprecated, a new one should be generated. This is done by repeating the actions described in Section 3.3, starting at step 3). Note that, except for the transient period when a temporary address is being regenerated, in normal operation at most one temporary address corresponding to a public address should be in a non-deprecated state

at any given time. Note that if a temporary address becomes deprecated as result of processing a Prefix Information Option with a zero Preferred Lifetime, then a new temporary address must not be generated. The Prefix Information Option will also deprecate the corresponding public address.

[0099] To insure that a preferred temporary address is always available, a new temporary address should be regenerated slightly before its predecessor is deprecated. This is to allow sufficient time to avoid race conditions in the case where generating a new temporary address is not instantaneous, such as when duplicate address detection must be run. It is recommended that an implementation start the address regeneration process REGEN_ADVANCE time units before a temporary address would actually be deprecated.

[0100] As an optional optimization, an implementation may wish to remove a deprecated temporary address that is not in use by applications or upper-layers. For TCP connections, such information is available in control blocks. For UDP-based applications, it may be the case that only the applications have knowledge about what addresses are actually in use. Consequently, one may need to use heuristics in deciding when an address is no longer in use (e.g., the default TEMP_VALID_LIFETIME suggested above).

3.5. Regeneration of Randomized Interface Identifiers

[0101] The frequency at which temporary addresses should change depends on how a device is being used (e.g., how frequently it initiates new communication) and the concerns of the end user. The most egregious privacy concerns appear to involve addresses used for long periods of time (weeks to months to years). The more frequently an address changes, the less feasible collecting or coordinating information keyed on interface identifiers becomes. Moreover, the cost of collecting information and attempting to correlate it based on interface identifiers will only be justified if enough addresses contain non-changing identifiers to make it worthwhile. Thus, having large numbers of clients

change their address on a daily or weekly basis is likely to be sufficient to alleviate most privacy concerns.

[0102] There are also client costs associated with having a large number of addresses associated with a node (e.g., in doing address lookups, the need to join many multicast groups, etc.). Thus, changing addresses frequently (e.g., every few minutes) may have performance implications.

[0103] This document recommends that implementations generate new temporary addresses on a periodic basis. This can be achieved automatically by generating a new randomized interface identifier at least once every (TEMP_PREFERRED_LIFETIME - REGEN_ADVANCE - DESYNC_FACTOR) time units.

[0104] As described above, generating a new temporary address REGEN_ADVANCE time units before a temporary address becomes deprecated produces addresses with a preferred lifetime no larger than TEMP_PREFERRED_LIFETIME. The value DESYNC_FACTOR is a random value (different for each client) that ensures that clients don't synchronize with each other and generate new addresses at exactly the same time. When the preferred lifetime expires, a new temporary address is generated using the new randomized interface identifier.

[0105] Because the precise frequency at which it is appropriate to generate new addresses varies from one environment to another, implementations should provide end users with the ability to change the frequency at which addresses are regenerated. The default value is given in TEMP_PREFERRED_LIFETIME and is one day. In addition, the exact time at which to invalidate a temporary address depends on how applications are used by end users. Thus the default value given of one week (TEMP_VALID_LIFETIME) may not be appropriate in all environments. Implementations should provide end users with the ability to override both of these default values.

[0106] Finally, when an interface connects to a new link, a new randomised interface identifier should be generated immediately together with a new set of temporary addresses. If a device moves from one Ethernet to another, generating a new set of temporary addresses from a different randomized interface identifier ensures that the device uses different randomized interface identifiers for the temporary addresses associated with the two links, making it more difficult to correlate addresses from the two different links as being from the same node.

4. Implications of Changing Interface Identifiers

[0107] The IPv6 addressing architecture goes to some lengths to ensure that interface identifiers are likely to be globally unique where easy to do so. During the IPng discussions of the GSE proposal [GSE], it was felt that keeping interface identifiers globally unique in practice might prove useful to future transport protocols. Usage of the algorithms in this document may complicate providing such a future flexibility.

[0108] The desires of protecting individual privacy vs. the desire to effectively maintain and debug a network can conflict with each other. Having clients use addresses that change over time will make it more difficult to track down and isolate operational problems. For example, when looking at packet traces, it could become more difficult to determine whether one is seeing behavior caused by a single errant machine, or by a number of them.

[0109] Some servers refuse to grant access to clients for which no DNS name exists. That is, they perform a DNS PTR query to determine the DNS name, and may then also perform an A query on the returned name to verify that the returned DNS name maps back into the address being used. Consequently, clients not properly registered in the DNS may be unable to access some services. As noted earlier, however, a node's DNS name (if non-changing) serves as a constant identifier. The wide deployment of the extension described in this document could challenge the practice of inverse-DNS-based

"authentication," which has little validity, though it is widely implemented. In order to meet server challenges, nodes could register temporary addresses in the DNS using random names (for example a string version of the random address itself).

[0110] Use of the extensions defined in this document may complicate debugging and other operational troubleshooting activities. Consequently, it may be site policy that temporary addresses should not be used. Implementations may provide a method for a trusted administrator to override the use of temporary addresses.

5. Defined Constants

[0111] Constants defined in this document include:

TEMP_VALID_LIFETIME -- Default value: 1 week. Users should be able to override the default value.

TEMP_PREFERRED_LIFETIME -- Default value: 1 day. Users should be able to override the default value.

REGEN_ADVANCE -- 5 seconds

MAX_DESYNC_FACTOR -- 10 minutes. Upper bound on DESYNC_FACTOR.

DESYNC_FACTOR -- A random value within the range 0 - MAX_DESYNC_FACTOR.

It is computed once at system start (rather than each time it is used) and must never be greater than (TEMP_VALID_LIFETIME - REGEN_ADVANCE).

6. Future Work

[0112] An implementation might want to keep track of which addresses are being used by upper layers so as to be able to remove a deprecated temporary address from internal data structures once no upper layer protocols are using it (but not before). This is in contrast to current approaches where addresses are removed from an interface when they become invalid [ADDRCONF], independent of whether or not upper layer protocols are still using them. For

TCP connections, such information is available in control blocks. For UDP-based applications, it may be the case that only the applications have knowledge about what addresses are actually in use. Consequently, an implementation generally will need to use heuristics in deciding when an address is no longer in use (e.g., as is suggested in Section 3.4).

[0113] The determination as to whether to use public vs. temporary addresses can in some cases only be made by an application. For example, some applications may always want to use temporary addresses, while others may want to use them only in some circumstances or not at all.

[0114] Suitable API extensions will likely need to be developed to enable individual applications to indicate with sufficient granularity their needs with regards to the use of temporary addresses.

7. Security Considerations

[0115] The motivation for this document stems from privacy concerns for individuals. This document does not appear to add any security issues beyond those already associated with stateless address autoconfiguration [ADDRCONF].

8. Acknowledgments

[0116] The authors would like to acknowledge the contributions of the IPNGWG working group and, in particular, Matt Crawford, Steve Deering and Allison Mankin for their detailed comments.

9. References

[0117] [ADDRARCH] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 2373, July 1998.

[0118] [ADDRCONF] Thomson, S. and T. Narten, "IPv6 Address Autoconfiguration", RFC 2462, December 1998.

[0119] [ADDR_SELECT] Draves, R. "Default Address Selection for IPv6", Work in Progress.

[0120] [COOKIES] Kristol, D. and L. Montulli, "HTTP State Management Mechanism", RFC 2965, October 2000.

[0121] [DHCP] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[0122] [DDNS] Vixie, R., Thomson, S., Rekhter, Y. and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[0123] [DISCOVERY] Narten, T., Nordmark, E. and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.

[0124] [GSE] Crawford, et al., "Separating Identifiers and Locators in Addresses: An Analysis of the GSE Proposal for IPv6", Work in Progress.

[0125] [IPSEC] Kent, S., Atkinson, R., "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[0126] [MD5] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[0127] [MOBILEIP] Perkins, C., "IP Mobility Support", RFC 2002, October 1996.

[0128] [RANDOM] Eastlake 3rd, D., Crocker S. and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[0129] [SERIALNUM] Moore, K., "Privacy Considerations for the Use of Hardware Serial Numbers in End-to-End Network Protocols", Work in Progress.

11. Full Copyright Statement

its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

[0132] Annex B is the draft IETF SIP-SEC-AGREEMENT document. This internet draft describes the Security-Client header that might be used to carry the value of prefix length between UE and proxy. According to this document, new "extension parameters" can be added to the header, like the one proposed in embodiments of the invention.

Annex B

[0133] Security Mechanism Agreement for SIP Sessions

Status of this memo

[0134] This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Abstract

[0135] SIP has a number of security mechanisms. Some of them have been built in to the SIP protocol, such as HTTP authentication or secure attachments. These mechanisms have even alternative algorithms and parameters. SIP does not currently provide any mechanism for selecting which security mechanisms to use between two entities. In particular, even if some mechanisms such as OPTIONS were used to make this selection, the selection would be vulnerable against the Bidding-Down attack. This document defines three header fields for negotiating the security mechanisms within SIP between a SIP entity and its next SIP hop. A SIP entity applying this mechanism must always require some

minimum security (i.e. integrity protection) from all communicating parties in order to secure the negotiation, but the negotiation can agree on which specific security is used.

1. Introduction

[0136] Traditionally, security protocols have included facilities to agree on the used mechanisms, algorithms, and other security parameters. The reason for this is that algorithm development typically uncovers problems in old algorithms and sometimes even produces new problems. Furthermore, different mechanisms and algorithms are suitable for different situations. Typically, protocols also select other parameters beyond algorithms at the same time.

[0137] The purpose of this specification is to define a similar negotiation functionality in SIP [1]. SIP has some security functionality built-in (e.g. HTTP Digest authentication [4]), it can utilize secure attachments (e.g. S/MIME [5]), and it can also use underlying security protocols (e.g. IPsec/IKE [2] or TLS [3]). Some of the built-in security functionality allows also alternative algorithms and other parameters. While some work within the SIP Working Group has been looking towards reducing the number of recommended security solutions (i.e., recommend just one lower layer security protocol), we can not expect to cut down the number of items in the whole list to one. There will still be multiple security solutions utilized by SIP. Furthermore, it is likely that new methods will appear in the future, to complement the methods that exist today.

[0138] Chapter 2 shows that without a secured method to choose between security mechanisms and/or their parameters, SIP is vulnerable to certain attacks. As the HTTP authentication RFC [4] points out, authentication and integrity protection using multiple alternative methods and algorithms is vulnerable to Man-in-the-Middle (MitM) attacks. More seriously, it is hard or sometimes even impossible to know whether a SIP peer entity is truly unable to perform (e.g., Digest, TLS, or S/MIME) or if a MitM attack is in action. In small networks consisting of workstations and servers these issues are not very

relevant, as the administrators can deploy appropriate software versions and set up policies for using exactly the right type of security. However, SIP will be deployed to hundreds of millions of small devices with little or no possibilities for coordinated security policies, let alone software upgrades, and this makes these issues much worse. This conclusion is also supported by the requirements from 3GPP [7].

[0139] Chapter 6 documents the proposed solution, and chapter 7 gives some demonstrative examples.

2. Problem Description

[0140] SIP has alternative security mechanisms such as HTTP authentication with integrity protection, lower layer security protocols, and S/MIME. It is likely that their use will continue in the future. SIP security is developing, and is likely to see also new solutions in the future.

[0141] Deployment of large number of SIP-based consumer devices such as 3GPP terminals requires all network devices to be able to accommodate past, current and future mechanisms; there is no possibility for instantaneous change since the new solutions are coming gradually in as new standards and product releases occur. It is sometimes even impossible to upgrade some of the devices without getting completely new hardware.

[0142] So, the basic security problem that such a large SIP-based network must consider, would be on how do security mechanisms get selected? It would be desirable to take advantage of new mechanisms as they become available in products.

[0143] Firstly, we need to know somehow what security should be applied, and preferably find this out without too many additional roundtrips. Secondly, selection of security mechanisms MUST be secure. Traditionally, all security protocols use a secure form of negotiation. For instance, after establishing mutual keys through Diffie-Hellman, IKE sends hashes of the previously sent

data -- including the offered crypto mechanisms. This allows the peers to detect if the initial, unprotected offers were tampered with.

[0144] The security implications of this are subtle, but do have a fundamental importance in building large networks that change over time. Given that the hashes are produced also using algorithms agreed in the first unprotected messages, one could ask what the difference in security really is. Assuming integrity protection is mandatory and only secure algorithms are used, we still need to prevent MitM attackers from modifying other parameters, such as whether encryption is provided or not. Let us first assume two peers capable of using both strong and weak security. If the initial offers are not protected in any way, any attacker can easily "downgrade" the offers by removing the strong options. This would force the two peers to use weak security between them. But if the offers are protected in some way -- such as by hashing, or repeating them later when the selected security is really on -- the situation is different. It would not be sufficient for the attacker to modify a single message. Instead, the attacker would have to modify both the offer message, as well as the message that contains the hash/repetition. More importantly, the attacker would have to forge the weak security that is present in the second message, and would have to do so in real time between the sent offers and the later messages. Otherwise, the peers would notice that the hash is incorrect. If the attacker is able to break the weak security, the security method and/or the algorithm should not be used.

[0145] In conclusion, the security difference is making a trivial attack possible versus demanding the attacker to break algorithms. An example of where this has a serious consequence is when a network is first deployed with integrity protection (such as HTTP Digest [4]), and then later new devices are added that support also encryption (such as S/MIME [1]). In this situation, an insecure negotiation procedure allows attackers to trivially force even new devices to use only integrity protection.

3. Solution

3.1 Requirements

[0146] The solution to the SIP security negotiation problem should have the following properties:

[0147] (a) It allows the selection of security mechanisms, such as lower layer security protocols or HTTP digest. It also allows the selection of individual algorithms and parameters when the security functions are integrated in SIP (such as in the case of HTTP authentication).

[0148] (b) It allows next-hop security negotiation.

[0149] (c) It is secure (i.e., prevents the bidding down attack.)

[0150] (d) It is capable of running without additional roundtrips. This is important in the cellular environment, where link delays are relatively high, and an additional roundtrip could delay the call set up further.

[0151] (e) It does not introduce any additional state to servers and proxies.

[0152] Currently, SIP does not have any mechanism which fulfills all the requirements above. The basic SIP features such as OPTIONS and Require, Supported headers are capable of informing peers about various capabilities including security mechanisms. However, the straight forward use of these features can not guarantee a secured agreement. HTTP Digest algorithm lists [4] are not secure for picking among the digest integrity algorithms, as is described in the HTTP authentication RFC [4] itself. More seriously, they have no provisions for allowing encryption to be negotiated. Hence, it would be hard to turn on possible future encryption schemes in a secure manner.

[0153] A self-describing security mechanism is a security mechanism that, when used, contains all necessary information about the method being used as well as all of its parameters such as algorithms. A non-self-describing security mechanism is a security mechanism that, when used, requires that the use of the method or some of its parameters have been agreed beforehand.

[0154] Most security mechanisms used with SIP are self-describing. The use of HTTP digest, as well as the chosen algorithm is visible from the HTTP authentication headers. The use of S/MIME is indicated by the MIME headers, and the CMS structures inside S/MIME describe the used algorithms. TLS is run on a separate port in SIP, and where IPsec/IKE is used, IKE negotiates all the necessary parameters.

[0155] The only exception to this list is the use of manually keyed IPsec. IPsec headers do not contain information about the used algorithms.

[0156] Furthermore, peers have to set up IPsec Security Associations before they can be used to receive traffic. In contrast S/MIME can be received even if no Security Association was in place, because the application can search for a Security Association (or create a new one) after having received a message that contains S/MIME.

[0157] In order to make it possible to negotiate both self-describing and non-self-describing security mechanisms, we need another requirement on the security agreement scheme:

[0158] (f) The security agreement scheme must allow both sides to decide on the desired security mechanism before it is actually used.

[0159] This decision can, and must, take place on both sides before we can be sure that the negotiation has not been tampered by a man-in-the-middle. This tampering will be detected later.

3.2. Overview of Operations

[0160] The message flow below illustrates how the mechanism defined in this document works:

1. Client ----------client list---------> Server
2. Client <---------server list---------- Server
3. Client ------(turn on security)------- Server

4. Client ----------server list---------> Server

5. Client <---------ok or error---------- Server

Figure 1: Security negotiation message flow

[0161] Step 1: Clients wishing to use this specification can send a list of their supported security mechanisms along the first request to the server.

[0162] Step 2: Servers wishing to use this specification can challenge the client to perform the security agreement procedure. The security mechanisms and parameters supported by the server are sent along in this challenge.

[0163] Step 3: The client then proceeds to select the highest-preference security mechanism they have in common and to turn on the selected security.

[0164] Step 4: The client contacts the server again, now using the selected security mechanism. The server's list of supported security mechanisms is returned as a response to the challenge.

[0165] Step 5: The server verifies its own list of security mechanisms in order to ensure that the original list had not been modified.

[0166] This procedure is stateless for servers (unless the used security mechanisms require the server to keep some state).

[0167] The client and the server lists are both static (i.e., they do not and cannot change based on the input from the other side). Nodes may, however, maintain several static lists, one for each interface, for example.

[0168] Between Steps 1 and 2, the server may set up a non-self-describing security mechanism if necessary. Note that with this type of security mechanisms, the server is necessarily stateful. The client would set up the non-self-describing security mechanism between Steps 2 and 4.

3.3. Syntax

[0169] We define three new SIP header fields, namely Security-Client, Security-Server and Security-Verify. Their BNF syntax is provided below:

```
security-client = "Security-Client" HCOLON
                  sec-mechanism *(COMMA sec-mechanism)
security-server = "Security-Server" HCOLON
                  sec-mechanism *(COMMA sec-mechanism)
security-verify = "Security-Verify" HCOLON
                  sec-mechanism *(COMMA sec-mechanism)
sec-mechanism   = mechanism-name *(SEMI mech-parameters)
mechanism-name  = ( "digest-integrity" / "tls" / "ipsec-ike" /
                  "ipsec-man" / "smime" / token )
mech-parameters = ( preference / algorithm / extension )
preference      = "q" EQUAL qvalue
qvalue          = ( "0" [ "." 0*3DIGIT ] )
                  / ( "1" [ "." 0*3("0") ] )
algorithm       = "alg" EQUAL token
extension       = generic-param
```

[0170] Note that qvalue is already defined in the SIP BNF [1]. We have copied its definitions here for completeness.

[0171] The parameters described by the BNF above have the following semantics:

[0172] Mechanism-name: It identifies the security mechanism supported by the client, when it appears in a Security-Client header fields, or by the server, when it appears in a Security-Server or in a Security-Verify header field. This specification defines five values:

[0173] - "tls" for TLS [3].

[0174] - "digest-integrity" for HTTP Digest [4] using additional integrity protection for the Security-Verify header field. The additional integrity protection consists of using the qop parameter to protect a MIME body (e.g., "message/sip") that contains the Security-Verify header field.

[0175] - "ipsec-ike" for IPsec with IKE [2].

[0176] - "ipsec-man" for manually keyed IPsec without IKE.

[0177] - "smime" for S/MIME [5].

[0178] Preference: The "q" value indicates a relative preference for the particular mechanism. The higher the value the more preferred the mechanism is. All the security mechanisms MUST have different "q" values. It is an error to provide two mechanisms with the same "q" value.

[0179] Algorithm: An optional algorithm field for those security mechanisms which are not self-describing or which are vulnerable for bidding-down attacks (e.g., HTTP Digest). In the case of HTTP Digest, the same rules apply as defined in RFC 2617 [4] for the "algorithm" field in HTTP Digest.

3.4. Protocol Operation

[0180] This section deals with the protocol details involved in the negotiation between a SIP entity and its next-hop SIP entity. Throughout the text the next-hop SIP entity is referred to as the first-hop proxy or outbound proxy. However, the reader should bear in mind that a user agent server can also be the next-hop for a proxy or, in absence of proxies, for a user agent client. Note as well that a proxy can also have an outbound proxy.

3.4.1 Client Initiated

[0181] A client wishing to establish some type of security with its first-hop proxy MUST add a Security-Client header field to a request addressed to this proxy (i.e., the destination of the request is the first-hop proxy). This header field contains a list of all the security mechanisms that the client supports. The client SHOULD NOT add preference parameters to this list. The client MUST add both a Require and Proxy-Require header field with the value "sec-agree" to its request.

[0182] The Security-Client header field is used by the server to include any necessary information in its response. For example, if digest-integrity is the chosen mechanism, the server includes an HTTP authentication challenge in the response. If S/MIME is chosen, the appropriate certificate is included.

[0183] A server receiving an unprotected request that contains a Require or Proxy-Require header field with the value "sec-agree" MUST challenge the client with a 494 (Security Agreement Required) response. The server MUST add a Security-Server header field to this response listing the security mechanisms that the server supports. The server MUST add its list to the response even if there are no common security mechanisms in the client's and server's lists. The serverÆs list MUST NOT depend on the contents of the client's list.

[0184] The server MUST compare the list received in the Security-Client header field with the list to be sent in the Security-Server header field. When the client receives this response, it will choose the common security mechanism with the highest "q" value. Therefore, the server MUST add the necessary information so that the client can initiate that mechanism (e.g., a WWW-Authenticate header field for digest-integrity).

[0185] When the client receives a response with a Security-Server header field, it MUST choose the security mechanism in the serverÆs list with the highest "q" value among all the mechanisms that are known to the client. Then, it MUST initiate that particular security mechanism as described in Section 3.5. This initiation may be carried out without involving any SIP message exchange (e.g., establishing a TLS connection).

[0186] If an attacker modified the Security-Client header field in the request, the server may not include in its response the information needed to establish the common security mechanism with the highest preference value (e.g., the WWW-authenticate header field is missing). A client detecting such a lack of information in the response MUST consider the current security negotiation process aborted,

and MAY try to start it again by sending a new request with a Security-Client header field as described above.

[0187] All the subsequent SIP requests sent by the client to that server SHOULD make use of the security mechanism initiated in the previous step. These requests MUST contain a Security-Verify header field that mirrors the serverÆs list received previously in the Security-Server header field. These requests MUST also have both a Require and Proxy-Require header fields with the value "sec-agree".

[0188] The server MUST check that the security mechanisms listed in the Security-Verify header field of incoming requests correspond to its static list of supported security mechanisms.

[0189] Note that, following the standard SIP header field comparison rules defined in [1], both lists have to contain the same security mechanisms in the same order to be considered equivalent. In addition, for each particular security mechanism, its parameters in both lists need to have the same values.

[0190] The server can proceed processing a particular request if, and only if, the list was not modified. If modification of the list is detected, the server MUST challenge the client with a 494 (Security Agreement Required). This response MUST include a challenge with server's unmodified list of supported security mechanisms. If the list was not modified, and the server is a proxy, it MUST remove the "sec-agree" value from both the Require and Proxy-Require header fields, and then remove the header fields if no values remain.

[0191] Once the security has been negotiated between two SIP entities, the same SIP entities MAY use the same security when communicating with each other in different SIP roles. For example, if a UAC and its outbound proxy negotiate some security, they may try to use the same security for incoming requests (i.e., the UA will be acting as a UAS).

[0192] The user of a UA SHOULD be informed about the results of the security mechanism negotiation. The user MAY decline to accept a particular security mechanism, and abort further SIP communications with the peer.

3.4.2 Server Initiated

[0193] A server decides to use the security negotiation described in this document based on local policy. A server that decides to use this negotiation MUST challenge unprotected requests regardless of the presence or the absence of any Require, Proxy-Require or Supported header fields in incoming requests.

[0194] A server that by policy requires the use of this specification and receives a request that does not have the sec-agree option tag in a Require, Proxy-Require or Supported header field MUST return a 421 (Extension Required) response. If the request had the sec-agree option tag in a Supported header field, it MUST return a 494 (Security Agreement Required) response. In both situation the server MUST also include in the response a Security-Server header field listing its capabilities and a Require header field with an option-tag "sec-agree" in it. All the Via header field entries in the response except the topmost value MUST be removed. This ensures that the previous hop is the one processing the response (see example in Section 5.3).

[0195] Clients that support the extension defined in this document MAY add a Supported header field with a value of "sec-agree". In addition to this, clients SHOULD add a Security-Client header field so that they can save a round trip in case the server decides to challenge the request.

3.5. Security mechanism initiation

[0196] Once the client chooses a security mechanism from the list received in the Security-Server header field from the server, it initiates that mechanism. Different mechanisms require different initiation procedures.

[0197] If TLS is chosen, the client uses the procedures of Section 8.1.2 of [1] to determine the URI to be used as an input to the DNS procedures of [6]. However, if the URI is a sip URI, it MUST treat the scheme as if it were sips, not sip. If the URI scheme is not sip, the request MUST be sent using TLS.

[0198] If digest-integrity is chosen, the 494 (Security Agreement Required) response will contain an HTTP Digest authentication challenge. The client MUST use the qop parameter to protect a MIME body (e.g., "message/sip") that contains the Security-Verify header field in the request. Currently, only the qop value ÆauthÆintÆ is able to provide required protection. Note that digest alone without placing Security-Verify header in the body would not fulfill the minimum security requirements of this specification.

[0199] To use "ipsec-ike", the client attempts to establish an IKE connection to the host part of the Request-URI in the first request to the server. If the IKE connection attempt fails, the agreement procedure MUST be considered to have failed, and MUST be terminated.

[0200] Note that "ipsec-man" will only work if the communicating SIP entities know which keys and other parameters to use. It is outside the scope of this specification to describe how this information can be made known to the peers.

[0201] In both IPsec-based mechanisms, it is expected that appropriate policy entries for protecting SIP have been configured or will be created before attempting to use the security agreement procedure, and that SIP communications use port numbers and addresses according to these policy entries. It is outside the scope of this specification to describe how this information can be made known to the peers, but it could be typically configured at the same time as the IKE credentials or manual SAs have been entered.

[0202] To use S/MIME, the client MUST construct its request using S/MIME. The client may have received the serverÆs certificate in an S/MIME body in the

494 (Security Agreement Required) response. Note that S/MIME can only be used if the next hop SIP entity is a UA.

3.6. Duration of Security Associations

[0203] Once a security mechanism has been negotiated, both the server and the client need to know until when it can be used. All the mechanisms described in this document have a different way to signal the end of a security association. When TLS is used, the termination of the connection indicates that a new negotiation is needed. IKE negotiates the duration of a security association. If the credentials provided by a client using digest-integrity are not longer valid, the server will re-challenge the client. It is assumed that when IPsec-man is used, the same out-of-band mechanism used to distribute keys is used to define the duration of the security association.

3.7. Summary of Header Field Use

[0204] The header fields defined in this document may be used to negotiate the security mechanisms between a UAC and other SIP entities including UAS, proxy, and registrar. Information about the use of headers in relation to SIP methods and proxy processing is summarized in Table 1.

Header field	where	proxy ACK BYE CAN INV OPT REG
Security-Client	R	ard - o - o o o
Security-Server	401,407,421,494	- o - o o o
Security-Verify	R	ard - o - o o oQQQQ
Header field	where	proxy SUB NOT PRK IFO UPD MSG
Security-Client	R	ard o o - o o o

Security-Server	401,407,421,494		o	o	-	o	o	o
Security-Verify	R	ard	o	o	-	o	o	o

Table 1: Summary of header usage.

[0205] The "where" column describes the request and response types in which the header field may be used. The header may not appear in other types of SIP messages. Values in the where column are:

[0206] - R: Header field may appear in requests.

[0207] - 401, 407 etc.: A numerical value or range indicates response codes with which the header field can be used.

[0208] The "proxy" column describes the operations a proxy may perform on a header field:

[0209] - a: A proxy can add or concatenate the header field if not present.

[0210] - r: A proxy must be able to read the header field, and thus this header field cannot be encrypted.

[0211] - d: A proxy can delete a header field value.

[0212] The next six columns relate to the presence of a header field in a method:

[0213] - o: The header field is optional.

4. Backwards Compatibility

[0214] A server that, by local policy, decides to use the negotiation mechanism defined in this document, will not accept requests from clients that do not support this extension. This obviously breaks interoperability with every plain SIP client. Therefore, this extension should be used in environments where it is somehow ensured that every client implements this extension. This extension may also be

used in environments where insecure communication is not acceptable if the option of not being able to communicate is also accepted.

5. Examples

[0215] The following examples illustrate the use of the mechanism defined above.

5.1. Client Initiated

[0216] A UA negotiates the security mechanism to be used with its outbound proxy without knowing beforehand which mechanisms the proxy supports.QQQQ

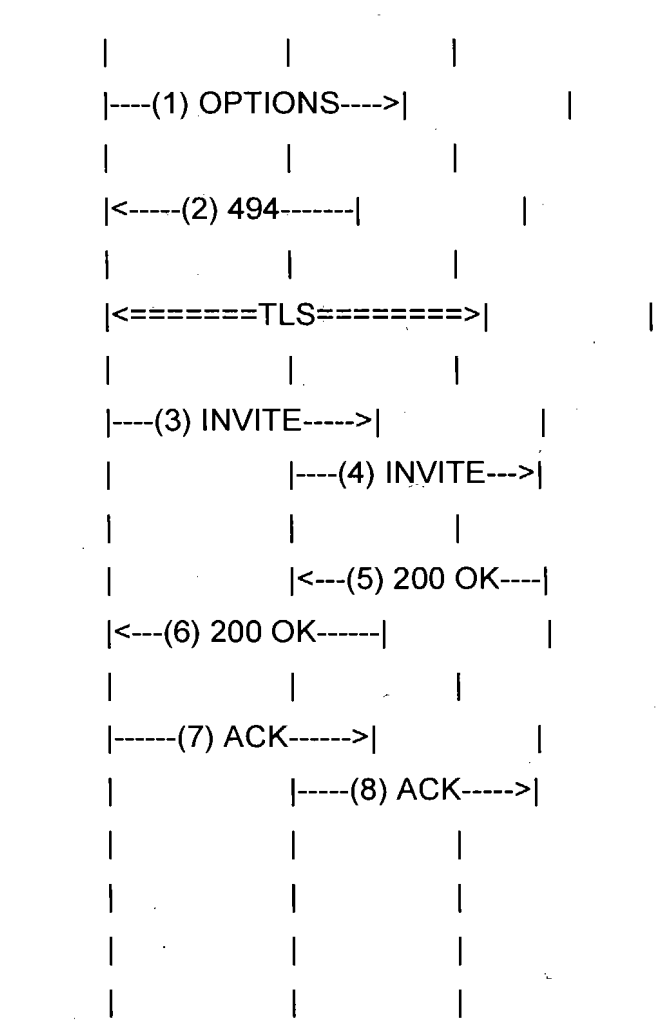


Figure 2: Negotiation initiated by the client

[0217] The UAC sends an OPTIONS request to its outbound proxy, indicating that it is able to negotiate security mechanisms and that it supports TLS and digest-integrity (Step 1 of figure 1). The outbound proxy challenges the UAC with its own list of security mechanisms û Ipsec and TLS (Step 2 of figure 1). The only common security mechanism is TLS, so they establish a TLS connection between them (Step 3 of figure 1). When the connection is successfully established, the UAC sends an INVITE over the TLS connection just established (Step 4 of figure 1). This INVITE contains the serverÆs security list. The server verifies it, and since it matches its static list, it processes the INVITE and forwards it to the next hop.

[0218] If this example was run without Security-Server header in Step 2, the UAC would not know what kind of security the other one supports, and would be forced to error-prone trials.

[0219] More seriously, if the Security-Verify was omitted in Step 4, the whole process would be prone for MitM attacks. An attacker could spoof "ICMP Port Unreachable" message on the trials, or remove the stronger security option from the header in Step 1, therefore substantially reducing the security.

[0220] (1) OPTIONS sip:proxy.example.com SIP/2.0

Security-Client: tls
Security-Client: digest-integrity
Require: sec-agree
Proxy-Require: sec-agree

[0221] (2) SIP/2.0 494 Security Agreement Required

Security-Server: ipsec-ike;q=0.1
Security-Server: tls;q=0.2

[0222] (3) INVITE sip:proxy.example.com SIP/2.0

Security-Verify: ipsec-ike;q=0.1
Security-Verify: tls;q=0.2

Route: sip:callee@domain.com

Require: sec-agree

Proxy-Require: sec-agree

[0223] The 200 OK response for the INVITE and the ACK are also sent over the TLS connection. The ACK (7) will contain the same Security-Verify header field as the INVITE (3).

5.2. Server Initiated

[0224] In this example of figure 3 the client sends an INVITE towards the callee using an outbound proxy. This INVITE does not contain any Require header field.

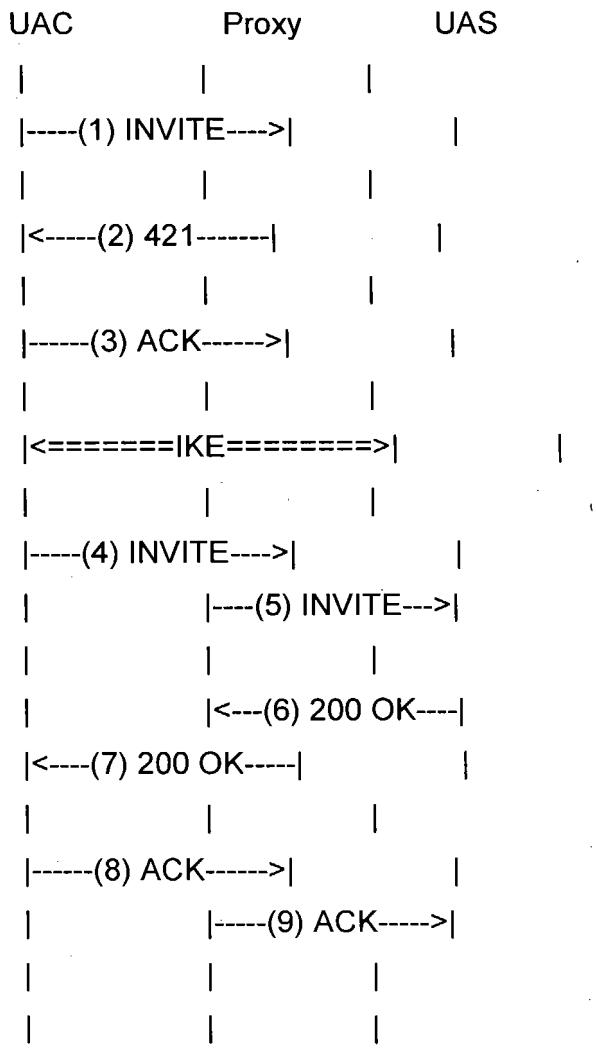
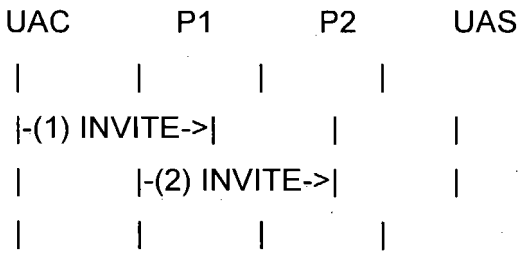


Figure 3: Server initiated security negotiation

[0225] The proxy, following its local policy, challenges the INVITE. It returns a 421 (Extension Required) with a Security-Server header field that lists IPsec-IKE and TLS. Since the UAC supports IPsec-IKE it performs the key exchange and establishes a security association with the proxy. The second INVITE (4) and the ACK (8) contain a Security-Verify header field that mirrors the Security-Server header field received in the 421. The INVITE (4), the 200 OK (7) and the ACK (8) are sent using the security association that has been established.

5.3 Security Negotiation between Proxies

[0226] The example in Figure 4 shows a security negotiation between two adjacent proxies. P1 forwards an INVITE (2) to P2. P2, by policy, requires that a security negotiation takes place before accepting any request. Therefore, it challenges P1 with a 421 (Extension Required) response (3). P2 removes all the Via entries except the topmost one (i.e., P1) so that P1 itself processes the response rather than forwarding it to the UAC. This 421 response contains a Security-Server header field listing the server's capabilities and a Require header field with an option-tag "sec-agree" in it. P2 includes "TLS" and "ipsec-ike" in the Security-Server header field. P1 sends an ACK (4) for the response and proceeds to establish a TLS connection, since this is the only security mechanism supported by P1. Once the TLS connection is established, session establishment proceeds normally. Messages (5), (8) and (11) are sent using the just established TLS connection. Messages (5) and (11) contain a Security-Verify header field that P2 removes before forwarding them to the UAS. Note that, following normal SIP procedures, P1 uses a different branch ID for INVITE (5) than the one it used for INVITE (2).



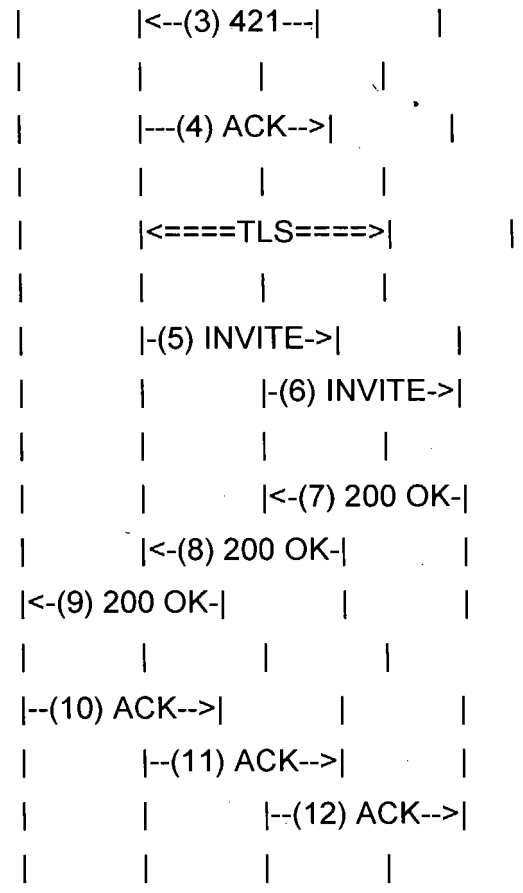


Figure 4: Negotiation between two proxies

6. Security Considerations

[0227] This specification is about making it possible to select between various SIP security mechanisms in a secure manner. In particular, the method presented here allow current networks using, for instance, Digest, to be securely upgraded to, for instance, IPsec without requiring a simultaneous modification in all equipment. The method presented in this specification is secure only if the weakest proposed mechanism offers at least integrity protection.

[0228] Attackers could try to modify the server's list of security mechanisms in the first response. This would be revealed to the server when the client returns the received list using the security.

[0229] Attackers could also try to modify the repeated list in the second request from the client. However, if the selected security mechanism uses encryption this

may not be possible, and if it uses integrity protection any modifications will be detected by the server.

[0230] Finally, attackers could try to modify the client's list of security mechanisms in the first message. The client selects the security mechanism based on its own knowledge of its own capabilities and the server's list, hence the client's choice would be unaffected by any such modification. However, the server's choice could still be affected as described below:

[0231] - If the modification affected the server's choice, the server and client would end up choosing different security mechanisms in Step 3 or 4 of figure 1. Since they would be unable to communicate to each other, this would be detected as a potential attack. The client would either retry or give up in this situation.

[0232] - If the modification did not affect the server's choice, there's no effect.

[0233] All clients that implement this specification MUST select HTTP Digest with integrity, TLS, IPsec, or any stronger method for the protection of the second request.

9. Normative References

[0234] [1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler "SIP: Session Initiation Protocol", Work in Progress, draft-ietf-sip-rfc2543bis-09.txt, IETF, February 2002.

[0235] [2] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, IETF, November 1998.

[0236] [3] T. Dierks, C. Allen, "The TLS Protocol Version 1.0", RFC 2246, IETF January 1999.

[0237] [4] Franks, J. et al, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, IETF, June 1999.

[0238] [5] B. Ramsdell and Ed, "S/MIME version 3 message specification", RFC 2633, IETF, June 1999.

[0239] [6] H. Schulzrinne and J. Rosenberg, "SIP: Locating SIP servers", Work in Progress, draft-ietf-sip-srv-06.txt, IETF, February 2002.